Monte Carlo Beam Search Decoder For Improving Quality of Automated Generated Stories

Zhaochen Luo

Fall 2020

Introduction

Natural language processing is a field within artificial intelligence which attempts to enable machines to process language like a human. Some of the most notable tasks within this field include machine translation, sentiment analysis, text summarization, and question answering. Many aspects of this field involve computer programs performing routine tasks with regard to text and intelligent agents have achieved human-like performances using deep learning. Almost all of these challenges are tackled by the use of recurrent neural networks(RNN) which is very useful in dealing with sequences of data. Automated story generation is among the many challenges under the natural language processing field. In automated story generation, the machine will try to generate a sequence of sentences that form a coherent story. Consistency and quality of each generated sentence are two important factors that will determine the quality of a generated story. This paper will focus on methods to improve the quality of each generated sentence.

Most work for story generation has been relying on the use of recurrent neural network, specifically the sequence to sequence model. In a sequence to sequence model, there is an encoder neural network that converts the input sentence into a fixed size state vector which is a matrix that contains information about the input sentence. The decoder neural network then takes the final state vector from the encoder and generate a sequence of words based on that input (Sutskever et al., 2014). In the decoding process within the decoder, beam search is often used (Freitag et al., 2017). In a normal search, one possible sentence is constructed by taking the more probable word from the recurrent neural network at each timestep. A beam search differs from a normal search in that at any given point in time, a number of possible partial sentences are kept instead of a single one. The partial sentences are called the beam and the number of possible partial sentences is called the beam size.

Using pure beam search has many limitations and drawbacks. When the model is not perfect and gives low probability to words that should have been selected, then

beam search will have no way of knowing to account for that and will never be able to include that word in the final output. Therefore, if there are desired words such as specific verbs that human would like to see in the output sentence that the model gives a low score on, those words might never appear in the output. In Martin's work with sentence eventification by transforming sentences into events, they were able to achieve consistent results with event to event network that predicts the next event from the current event where each event is a state tuple consisting of subject, verb, object, and a modifier term (Martin et al., 2017). However, their recurrent neural network does not produce quality results when outputting a sentence from an input event because of the limitations of pure beam search. Currently, they have not found a good way to dynamically make sure the input events appear in the output while also improving good sentence readability.

In Martin's work, their event to sentence network uses a beam search decoder. In addition, their event representation explicitly states which words should be in the output sentence (Martin et al., 2017). All the information the output sentence should include is included in the words in the event, and the role of the event to sentence neural network should be to reconstruct the input event in a meaningful way. As mentioned before, traditional beam search might not include certain words at all if they process the direct output from the model. In this study, translating events to sentence using a Monte Carlo beam decoder will be explored. The name Monte Carlo beam search is based on the idea of Monte Carlo tree search which relies on simulation. In Monte Carlo beam search, the beams which contain sentences will be reweighted based on if they contain certain words from the input event. At each timestep of a prediction, a simulation will be done from a given sentence in a beam and the corresponding sentence will be rescored based on the score of the simulation which is computed as how adherent the output is to the input (Cazenave, 2012). The scoring function will give a high score to a sentence with high matching. This will allow a sentence with high matching that might have low raw probability score to become the top candidate. This method aims to fix the inconsistency of traditional recurrent neural

networks by adding in another scoring term into the output calculation. In addition, by using the input as the constraint, this method could work well for tasks that require the output to include elements from the input.

Literature Review

As is the case with many other natural language processing tasks, most of the story generation projects have been based on sequence to sequence networks. However, story generation is different from some of them because story generation requires a much longer dependency and is much more difficult to ensure consistency and meaningfulness. There have been many variants of story generation projects that introduce new input in order to guide the story (Sutskever et al., 2014). With added information through those inputs, the neural network can base its prediction on both its previous generation as well as the new forms of inputs provided. Jain used the chaining of small short story descriptions to form a large story (Jain et al., 2017). Fan used a hierarchical story generation. Their model generates a prompt and then generates the rest of the story by conditioning on the prompt (Fan et al., 2018). This allowed them to generate stories that do not drift off topic which is a very common problem in story generation. Another representation of events is explored by Martin where the story generation is done in a unit of events. These events are then interpreted back into human readable sentences (Martin et al., 2017). This would be the architecture this paper strives to improve upon.

In most of these cases, the encoder and decoder network are chosen to be recurrent neural networks because they can handle variable length input much better. However, convolutional neural networks have also been used as encoder and decoder to utilize the parallelism of graphics processing units(GPUs) (Gehring et al. 2017). During the decoding process, a beam decoder is often used. A greedy decoder is one that always picks the word with the highest probability given by the recurrent neural network at each timestep. This could lead to a suboptimal solution because the optimal

solution could have a low score in the beginning and greedy decoder would never pick it (Germann et al., 2001). The most ideal way to generate the sentence with the highest overall probability would be to generate all the possible sentences and then pick out the best sentence. However, this would be computationally impossible because the number of possible sentences grows exponentially with the increase of length. Therefore, a beam search decoder is often used to approximate that.

In a beam search decoder, the decoder will keep track of k possible sequences where k is often set to be 5 or 10. At the next timestep, k possible next tokens are generated from each sequence and then the top k scoring sequences overall are kept (Freitag et al., 2017). This increases the probability of getting a more optimal output by keeping track of more hypotheses. A beam search differs from a normal search in that at any given point in time, a number of possible partial sentences are kept instead of a single one. The partial sentences are called the beam and the number of possible partial sentences is called the beam size. However, traditional beam search still suffers from lack of diversity. Even though a beam size k number of sentences are kept, most of them are going to look similar and only differ by a few words. Since the only way for a sequence with a low score to stay in the beam is to have a word that has a high score in the next timestep, the output is highly dependent on the decoder neural net model. This does not utilize the full benefit of a beam decoder and would not approximate the probability of the output sentence well. Studies have been done to make each beam more diverse by incorporating a diverse factor that would penalize two sentences that are really similar (Vijayakumar et al., 2016). One flaw of this approach is that it does not adhere to the input. This model would make the beam very diverse but each of them might not match up too well with the input. Since story generation often requires consistency, creativity and quality, such limitations could reduce the interestingness of the generated stories. In our setup, the model converts an event into a sentence. Therefore, it is important to generate outputs that follow the words in the input event. Currently, there has not been any work to our knowledge that tries to improve on traditional beam search decoders while forcing them to adhere to the input.

Methods

Since the main job of the Event-to-Sentence network is to convert an event into a human-readable sentence, it is important for the generated sentence to adhere as much as possible to the input event. In other words, the generated sentence should contain as many words from the input event as possible while maintaining good sentence structure in the best case. The Monte Carlo tree search algorithm can assign an alternative score based on how the generated sentence adhere to the input event. The name Monte Carlo tree search comes from the simulation nature of the algorithm. The algorithm bases its decision on not only the current state it sees but also the score coming from the future by generating a complete sentence from a partial sentence.

The overall architecture of this method is the same as a standard sequence-to-sequence network as there is an encoder and a decoder and the only difference lies in the decoder. The Monte Carlo tree search will be implemented in the decoder. The Monte Carlo tree search will keep track of k possible partial sentences at each time step. At each time step, all possible next words from those k partial sentences are generated and the top k + m words that have the highest probability from the decoder recurrent neural network are kept where k=5 and n=3 in this experiment. Monte Carlo playouts are done from each of the k + m partial sentences. A playout refers to using the decoder to generate new words for a sentence until that sentence reaches a “EOS” which stands for “End of Sentence” and is usually used to signify that a sentence has ended. In this playout, one word will be generated for one sentence at each time step by the decoder RNN. After the sentence has reached “EOS” and ended, this sentence will be compared to the input event to see how similar they are.
The metrics used in this experiment is a weighted Bilingual Evaluation Understudy (BLEU) score. BLEU score is a metric that is used to evaluate how much overlap a generated sentence has with a reference sentence. When computing the BLEU score, it will compute n-gram similarities from n=1 to n=5 where n-gram is a contiguous

sequence of n words. When computing n-gram similarity, the score will be based on whether that word is in the generated sentence. In this approach, the input reference is a 5-tuple consisting of verb, subject, object, preposition and a modifier. There are 5 weights corresponding to how important each of these items are to the whole sentence. Ideally all 5 items should be included in the generated sentence, every time an item is missing from the final generated sentence, the weight for that word is bumped up through where a=0.99. This update equation will first give the new score as a fraction of the old score. It will then give additional weight to items that required a score boost. When the weight is higher for an item, the penalty for excluding that item in the generated sentence will be higher. Therefore, this encourages the model to try to generate the corresponding item in the next round since that item will make up for more of the final score than the other items. This reweighting procedure is done on the training dataset and the weights will be fixed for testing time. The score that will be used to score a sentence is calculated with standard BLEU score which is calculated from 1-gram to 5-grams and 1-gram BLEU score for each of the items as reference. The resulting score will be an average of these scores as . The score for the new word plus the old partial sentence will be computed as . This score will be used to determine how closely the generated sentence constructs a sentence using the input 5-tuple.
Playouts will be run on a sentence until it has reached “EOS” and the whole generation process stops when all k + m sentences have reached “EOS” and ended. The sentence with the best final score will be kept as the final generated sentence for the current input event. Using Monte Carlo tree search within the decoder allows for alternative scoring metrics that can offset some randomness from the recurrent neural network.

Results/Discussion

Table 1: Perplexity and BLEU score

Model	**BLEU**	**Length**

Baseline Sequence-to-Sequence	0.0425	6.59
Monte Carlo Beam Decoder	0.0453	7.91

The results of the event-to-sentence network is shown in Table 1. The BLEU score is the main metric used to evaluate performance of generated sentences for the two different methods. From the table, it shows that the Monte Carlo Beam Decoder does produce results that have high BLEU scores meaning the results align more with the input events. Although the gap is small, the BLEU score only ranges at a small value in this setup with story generation and is naturally low. Helping improve the BLEU score and guiding the model to produce sentences that match words in the input event more are the main reasons the Monte Carlo Beam Decoder is used in this project. Using Monte Carlo Tree Search and the BLEU score function to score each sentence gives the model an ability to look ahead by running simulations of what the whole sentence might generate into. This gives the model a way to use future data to score the current word which makes the scoring more reliable. With the case of a plain baseline sequence-to-sequence model, the model only has access to the raw probability and uses that as the only information to pick best words at each time step. Therefore, the baseline model only has limited information at each decision step and small changes in the output probability scores can have a large impact on the sentences that are generated. This is an important reason behind the inconsistent results produced from the baseline model. The Monte Carlo Tree Search that gets run at each time step reduces the effect of this problem by incorporating the BLEU score evaluation into the decision making. Although the probability might not be stable across runs, the BLEU score will give words that can lead to sentences with higher BLEU scores high scores on average which gives the model more stability and consistency to counter the randomness in the neural network.

Table 2: Example output sentences by Monte Carlo Beam Decoder

Input Event	**Monte Carlo Beam Decoder Output**
(<PRP>, act-114-1-1, to, ∅, event.n.01)	<PRP> moves to the nearest natural.object.n.01.
(<PERSON>2, send-11.1, through, <PERSON>6, <LOCATION>1)	<PERSON>2 passes this undercover in the body_part.n.01 and collapses.
(<PERSON>0, admire-31.2, ∅, <PERSON>3, ∅)	<PERSON>0 hates <PERSON>3 saying <PRP> not ready for duration.n.0.3

Table 2 shows some sample outputs generated by the network using the Monte Carlo Tree Search in the decoder. The input events contain generalized vocabularies. For example, act-114-1-1 in the first sample is a generalized term for the word “move” to abstract away the diverse vocabularies the model needs to know about. This groups similar words together and the generalization allows the model to learn better. As the samples show, the Monte Carlo Beam Decoder is successful at generating sentences that contain target words in the input event. Although it does not guarantee every word in the input event will be included, it is effective at including the most important words such as the subject and verbs because the model panelizes missing those words more when computing the BLEU score. Overall, the Monte Carlo Beam Decoder has shown to improve the BLEU score of generated sentences and improve the consistency of the results.

References

Angela Fan, Yann Dauphin, and Mike Lewis. 2018. Hierarchical neural story generation. In Conference of the Association for Computational Linguistics (ACL).

A. Vijayakumar, M. Cogswell, R. R. Selvaraju, Q. Sun, S. Lee, D. Crandall, and D. Batra. Diverse beam search: Decoding diverse solutions from neural sequence models. In arXiv:1610.02424, 2016.

Cazenave, T. 2012. Monte carlo beam search. Computational Intelligence and AI in Games, IEEE Transactions on 4(1):68–72.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann Dauphin. 2017. Convolutional sequence to sequence learning.

Lara J Martin, Prithviraj Ammanabrolu, William Hancock, Shruti Singh, Brent Harrison, and Mark O Riedl. 2017. Event representations for automated story generation with deep neural nets. arXiv preprint arXiv:1706.01331.

Markus Freitag and Yaser Al-Onaizan, “Beam search strategies for neural machine translation,” arXiv preprint arXiv:1702.01806, 2017.

P. Anderson, B. Fernando, M. Johnson, and S. Gould. Guided open vocabulary image captioning with constrained beam search. Empirical Methods in Natural Language Processing (EMNLP), 2017.

Parag Jain, Priyanka Agrawal, Abhijit Mishra, Mohak Sukhwani, Anirban Laha, and Karthik Sankaranarayanan. 2017. Story generation from sequence of independent short descriptions. arXiv preprint arXiv:1707.05501.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In Advances in neural information processing systems, 3104–3112.

Ulrich Germann, Mike Jahr, Kevin Knight, Daniel Marcu, and Kenji Yamada. 2001. Fast decoding and optimal decoding for machine translation. In Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL’01), pages 228–235, Toulouse, France, July 6–11.